〈[파이썬의 기본 보물들] 제1장: 이름표 붙은 보관함, 딕셔너리!〉

선생님: (여러 개의 이름표가 붙은 서랍장을 가리키며) 얘들아, 안녕! 우리가 리스트나 튜플은 물건을 순서대로 착착 넣어두는 긴 상자 같았지? 오늘 배울 **딕셔너리**는 이 서랍장처럼, 각 칸마다 '이름표(키, Key)'를 붙여두고, 그 이름표에 해당하는 물건(값, Value)을 쏙쏙 보관하는 아주 똑똑한 보관함이란다!

코딩새싹 🌱: 이름표가 붙은 보관함이요? 그럼 순서 번호 대신 이름표로 물건을 찾는 거예요?

선생님: 바로 그거야! 딕셔너리는 중괄호 {}를 사용해서 만들고, 각 이름표(키)와 물건(값)은 콜론(:)으로 짝을 지어준단다. 중요한 건, 이름표(키)는 절대로 중복될 수 없어! 각 서랍의 이름은 유일해야 하니까.

```
# 딕셔너리 만들기: {'키1': 값1, '키2': 값2, ...}
my_dict = {'name': '왕새우', 'age': 25, 'score': 100}
print("나의 첫 딕셔너리:", my_dict)
print("딕셔너리의 타입은?", type(my_dict)) # 〈class 'dict'〉

# 이 보관함에는 몇 개의 이름표-물건 쌍이 들어있을까?
print("보관함 칸 수:", len(my_dict))
```

〈1. 이름표로 보물 찾기! (값에 접근하기)〉

선생님: 자, 이 `my_dict` 보관함에서 'name'이라는 이름표가 붙은 칸에는 뭐가 들어있을까? 대괄호 [] 안에 키를 써서 확인할 수 있어!

```
print("이름표 'name'의 내용물:", my_dict['name'])  # 왕새우
print("이름표 'age'의 내용물:", my_dict['age'])    # 25
```

궁금이 💡: 만약 없는 이름표를 부르면 어떻게 돼요? 예를 들어 'school'이라는 이름표는 없는데요!

선생님: 아주 예리한 질문이야! 존재하지 않는 키로 찾으려고 하면, "그런 이름표는 없어!" 하고 `KeyError`라는 에러가 발생한단다.
그래서 더 안전하게 물건을 찾는 방법이 있어. 바로 `.get()` 메소드지!

```
# print(my_dict['school']) # 이 코드는 KeyError 발생!

# .get() 메소드로 안전하게 찾기
school_value = my_dict.get('school') # 'school' 이름표가 있으면 내용물을, 없으면 None을 돌려줘.
print("get('school') 결과:", school_value) # None (아무것도 없다는 뜻)

# .get()은 두 번째 재료로 "만약 없으면 이걸 대신 줘!" 하는 기본값을 설정할 수도 있어.
hometown_value = my_dict.get('hometown', "정보 없음")
print("get('hometown', '정보 없음') 결과:", hometown_value)
```

코딩새싹 🌱: 와! `.get()`을 쓰니까 에러 없이 안전하게 확인할 수 있네요!

〈2. 보관함 내용물 바꾸고, 새 이름표도 붙이기! (값 수정 및 추가)〉

선생님: 딕셔너리 보관함의 좋은 점은 내용물을 언제든지 바꿀 수 있다는 거야!

```
my_dict['name'] = '별난새우' # 'name' 이름표의 내용물을 '별난새우'로 변경!
my_dict['age'] = 100      # 'age' 내용물도 100으로 변경!
print("\n내용물 변경 후:", my_dict)
```

선생님: 그리고 새로운 이름표를 붙여서 새 물건을 보관할 수도 있어!

```
# 'school'이라는 이름표가 있는지 먼저 확인해볼까? ( \b 는 오타일 수 있으니 'school'로!)
if not 'school' in my_dict: # 'school' 키가 my_dict 안에 없다면,
  my_dict['school'] = "대단한대학교" # 'school' 이름표로 "대단한대학교"를 새로 보관!
print("새 이름표 추가 후:", my_dict)
```

궁금이 💡: in 연산자로 키가 있는지 없는지 확인할 수 있군요! 그리고 없는 키에 값을 넣으면 그냥 새로 생기는 거네요!

<3. 보관함 정리하기! (키-값 쌍 삭제하기)>

선생님: 때로는 보관함에서 필요 없는 물건을 빼내야 할 때도 있겠지? del 키워드나 .pop() 메소드를 사용하면 돼.

```
# del my_dict['score'] # 'score' 이름표와 내용물을 함께 삭제! (주석 해제하고 실행해봐!)
# print("del 후:", my_dict)

# .pop()은 물건을 꺼내서 우리에게 보여주고, 그 칸은 비워버려!
# 마치 리스트의 .pop()과 비슷하지만, 딕셔너리는 "키"를 알려줘야 해!
removed_score = my_dict.pop('score') # 'score' 칸에서 내용물을 꺼내 removed_score에 저장!
print(f"\n.pop('score')로 꺼낸 값: {removed_score}")
print("pop 후 딕셔너리:", my_dict)
```

코딩새싹 🌱: list.pop()은 맨 뒤에 걸 꺼내거나 번호로 꺼냈는데, dict.pop()은 이름표(키)로 꺼내는군요!

<4. 보관함 전체 둘러보기! (.keys(), .values(), .items())>

선생님: 우리 보관함에 어떤 이름표들이 있는지, 어떤 물건들이 들어있는지, 또는 이름표와 물건을 쌍으로 한 번에 보고 싶을 때 쓰는 아주 유용한 기술들이 있어!

```
# 현재 my_dict 상태: {'name': '별난새우', 'age': 100, 'school': '대단한대학교'}

# .keys() : 모든 이름표(키)들만 모아서 보여줘!
print("\n모든 키들:", list(my_dict.keys())) # list()로 감싸야 보기 편해!

# .values() : 모든 물건(값)들만 모아서 보여줘!
print("모든 값들:", list(my_dict.values()))
print("값들만 하나씩 출력해볼까?")
for value in my_dict.values():
  print(value)
```

```
# .items() : (이름표, 물건) 짝꿍들을 모아서 보여줘!
print("\n(키, 값) 짝꿍들:", list(my_dict.items()))
print("짝꿍들을 하나씩 출력해볼까?")
for key, val in my_dict.items(): # 튜플 언패킹 기억나지? (key, val) 형태로!
  print(f"키: {key}, 값: {val}")
```

궁금이 💡: my_dict.items()로 반복하니까 key랑 val을 한 번에 받을 수 있어서 정말 편하네요! for문이랑 찰떡궁합 같아요!

<5. 실전! 딕셔너리로 문장 만들기!>

선생님: 자, 이제 딕셔너리에 담긴 정보를 이용해서 멋진 문장을 만들어볼까?

```
person = {"name": "Alice", "city": "San Francisco", "email": "alice@email.com"}

# person 딕셔너리의 값을 이용해서 문장 완성하기!
print(f"\n결과: {person['name']}가 사는 곳은 {person['city']}이고, 이메일은 {person['email']}입니다.")
```

코딩새싹 🌱: 와! f-string 안에 person['name'] 이렇게 쓰니까 바로바로 정보를 가져와서 문장을 만들 수 있네요! 정말 편리해요!

선생님: (서랍장을 닫으며) 아주 잘했어! 오늘 우리는 이름표(키)와 내용물(값)로 이루어진 똑똑한 보관함, 딕셔너리에 대해 배웠어. 순서보다는 '이름표'로 빠르게 정보를 찾고 싶을 때 아주 유용하지. 이 딕셔너리라는 보물 상자를 잘 활용하면, 앞으로 훨씬 더 체계적이고 강력한 프로그램을 만들 수 있을 거란다! 오늘 탐험도 정말 훌륭했어! 🎁👍